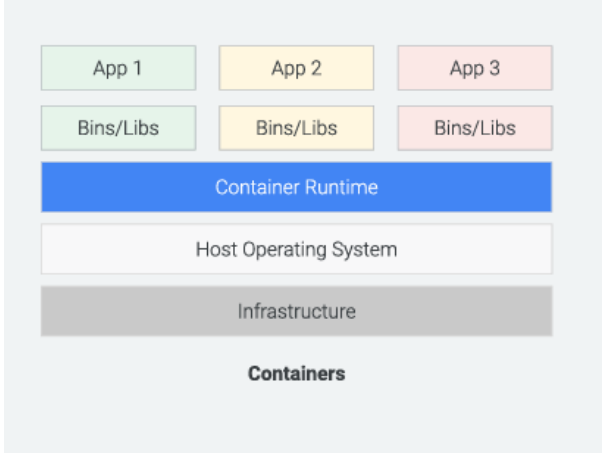


**U.S. Patent No. 7,784,058 (“’058 Patent”)**

Accused Instrumentalities: Google’s “Migrate to Containers,” and all versions and variations thereof since the issuance of the asserted patent.

**Claim 1**

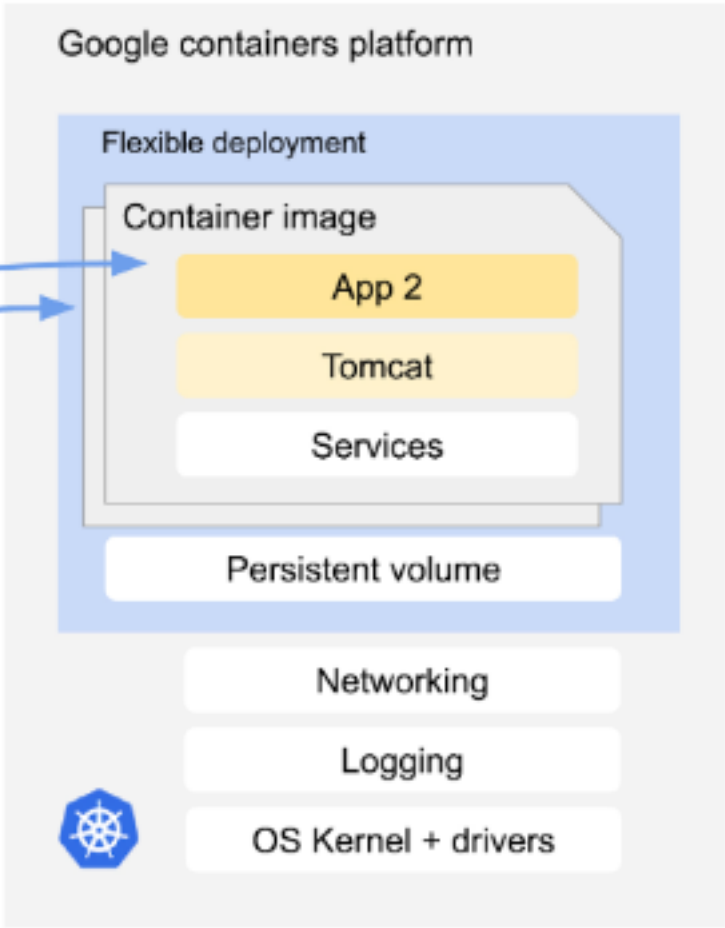
Claim 1	Accused Instrumentalities
<p>[1pre] 1. A computing system for executing a plurality of software applications comprising:</p>	<p>To the extent the preamble is limiting, each Accused Instrumentality comprises or constitutes a computing system for executing a plurality of software applications as claimed.</p> <p><i>See claim limitations below.</i></p> <p><i>See also, e.g.:</i></p> <p>Use Migrate to Containers to modernize traditional applications away from virtual machine (VM) instances and into native containers that run on Google Kubernetes Engine (GKE), GKE Enterprise clusters, or Cloud Run platform. You can migrate workloads from VMs that run on VMware or Compute Engine, giving you the flexibility to containerize your existing workloads with ease. Migrate to Containers supports modernization of IBM WebSphere, JBoss, Apache, Tomcat, WordPress, Windows IIS applications, as well as containerisation of Linux-based applications.</p> <p><a href="https://cloud.google.com/migrate/containers/docs/getting-started">https://cloud.google.com/migrate/containers/docs/getting-started</a>.</p> <p>A container is a way of packaging a given application’s code and dependencies so that the application will run easily in any computing environment. This solves the common problem of</p>

Claim 1	Accused Instrumentalities
	<div data-bbox="632 261 1230 711"><p>The diagram illustrates a container architecture stack. At the top, three application boxes labeled 'App 1' (green), 'App 2' (yellow), and 'App 3' (pink) are shown. Below each app is a corresponding 'Bins/Libs' box in the same color. These three pairs are stacked on top of a single blue box labeled 'Container Runtime'. This runtime sits on a white box labeled 'Host Operating System', which in turn sits on a grey box labeled 'Infrastructure'. The entire stack is enclosed in a light grey box with the label 'Containers' at the bottom center.</p></div> <p data-bbox="632 735 1629 769"><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p> <p data-bbox="659 813 1188 1240">Containers can run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data centers on-premises; and of course, in the public cloud.</p>

Claim 1	Accused Instrumentalities
	<p>Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services.</p> <p><a href="https://cloud.google.com/learn/what-are-containers">https://cloud.google.com/learn/what-are-containers</a></p>
[1a] a) a processor;	<p>Each Accused Instrumentality comprises a processor.</p> <p><i>See, e.g.:</i></p> <p>Containers virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications.</p> <p><a href="https://cloud.google.com/learn/what-are-containers">https://cloud.google.com/learn/what-are-containers</a></p> <ul style="list-style-type: none"> <li>• <b>Higher utilization and density</b>, leveraging automatic bin-packing and auto-scaling capabilities, Kubernetes places containers optimally in nodes based on required resources while scaling as needed, without impairing availability. In addition, unlike VMs, all containers on a single node share one copy of the operating system and don't each require their own OS image and vCPU, resulting in a much smaller memory footprint and CPU needs. This means more workloads running on fewer compute resources.</li> </ul>

Claim 1	Accused Instrumentalities
	<p><a href="https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration">https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration</a></p> <p>Containers use specific features of the Linux kernel that “trick” individual applications into thinking they’re in their own unique environment, even though multiple applications share the same host kernel. (If you’re not familiar with the Linux kernel, it’s a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests).</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>
<p>[1b] b) an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,</p>	<p>Each Accused Instrumentality comprises an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor.</p> <p><i>See, e.g.:</i></p> <ul style="list-style-type: none"> <li>• Containers are much more lightweight than VMs</li> <li>• Containers virtualize at the OS level while VMs virtualize at the hardware level</li> <li>• Containers share the OS kernel and use a fraction of the memory VMs require</li> </ul> <p><a href="https://cloud.google.com/learn/what-are-containers">https://cloud.google.com/learn/what-are-containers</a></p>

Claim 1	Accused Instrumentalities
	<p><b>Kernel mode</b></p> <p>Kernel mode refers to the processor mode that enables software to have full and unrestricted access to the system and its resources. The OS kernel and kernel drivers, such as the file system driver, are loaded into protected memory space and operate in this highly privileged kernel mode.</p> <p><a href="https://www.techtarget.com/searchdatacenter/definition/kernel">https://www.techtarget.com/searchdatacenter/definition/kernel</a></p>

Claim 1	Accused Instrumentalities
	<div data-bbox="682 276 1402 1198"><p>The diagram illustrates the Google containers platform architecture. It is a layered stack. At the top is the 'Google containers platform' layer. Below it is the 'Flexible deployment' layer, which contains a 'Container image' box. The 'Container image' box is divided into three sections: 'App 2' (yellow), 'Tomcat' (yellow), and 'Services' (white). Below the 'Container image' box is a 'Persistent volume' box. Below that is a 'Networking' box. Below that is a 'Logging' box. At the bottom is the 'OS Kernel + drivers' box. A blue Kubernetes logo is located to the left of the 'OS Kernel + drivers' box. Two blue arrows point from the left towards the 'Container image' box, indicating input or deployment.</p></div> <p data-bbox="632 1235 1860 1305"><a href="https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line">https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line</a></p>

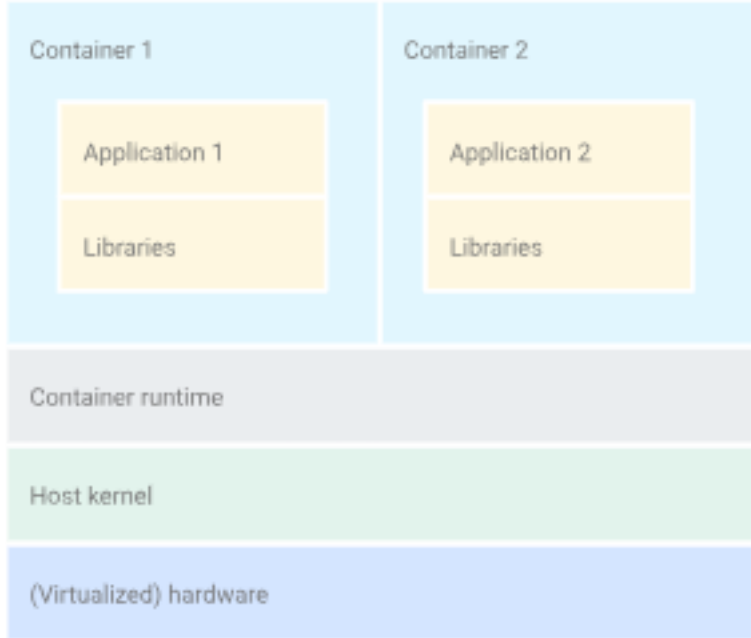
Claim 1	Accused Instrumentalities
	<p>The migration prerequisites are dependent on your specific migration environment. Confirm that your workloads' OS and source platform are compatible for migration by reviewing the prerequisites for your specific migration environment:</p> <p><a href="https://cloud.google.com/migrate/containers/docs/setting-up-overview">https://cloud.google.com/migrate/containers/docs/setting-up-overview</a></p> <p>Containers use specific features of the Linux kernel that “trick” individual applications into thinking they’re in their own unique environment, even though multiple applications share the same host kernel. (If you’re not familiar with the Linux kernel, it’s a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests).</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>
[1c] c) a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and	<p>Each Accused Instrumentality comprises a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode.</p> <p><i>See, e.g.:</i></p>

Claim 1	Accused Instrumentalities
	<p>A “container image” is your application and its dependencies, and uses a “base image” as the basis for the container image</p> <p>The container image specifies the container’s file system. For example, if you’re running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or <b>base image</b>. This base image is the basis of your container, so you’ll want to ensure that it’s properly patched and free from known vulnerabilities.</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>

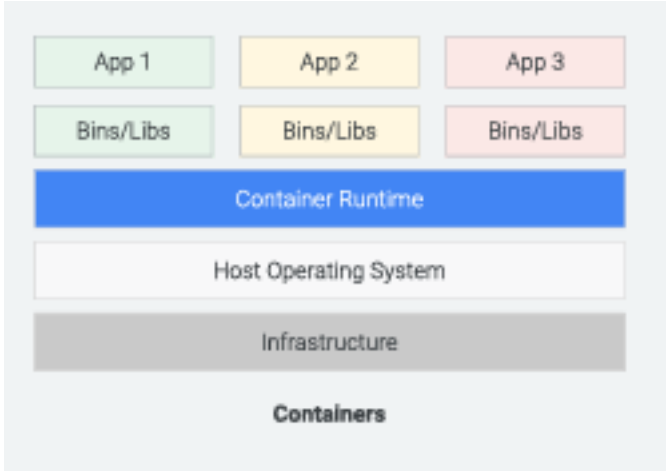


Claim 1	Accused Instrumentalities
	<p>A base image is the starting point for most container-based development workflows. Developers start with a base image and layer on top of it the necessary libraries, binaries, and configuration files used to run their application.</p> <p>Many base images are basic or minimal Linux distributions: Debian, Ubuntu, Red Hat Enterprise Linux (RHEL), Rocky Linux, or Alpine. Developers can consume these images directly from Docker Hub or other sources. There are official providers along with a wide variety of other downstream repackagers that layer software to meet customer needs.</p> <p>Google maintains base images for building its own applications. These images are built from the same source that Docker Hub uses. Therefore, they match the images you would get from Docker Hub.</p> <p><a href="https://cloud.google.com/software-supply-chain-security/docs/base-images">https://cloud.google.com/software-supply-chain-security/docs/base-images</a></p> <p>The <a href="#">preconfigured base images</a> provided by Cloud Workstations contain only a minimal environment with IDE, basic Linux terminal and language tools and a <code>sshd</code> server. To expedite the environment setup of specific development use cases, you can create custom container images that extend these base images to pre-install tools and dependencies and that run automation scripts.</p> <p>For custom container images, we recommend setting up a pipeline to automatically rebuild these images when the Cloud Workstations base image is updated, in addition to running a container scanning tool such as <a href="#">Artifact Analysis</a> to inspect any additional dependencies you added. You're responsible for maintaining and updating custom packages and dependencies added to custom images.</p> <p><a href="https://cloud.google.com/workstations/docs/customize-container-images">https://cloud.google.com/workstations/docs/customize-container-images</a></p> <p>A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of</p>


Claim 1	Accused Instrumentalities
	<p>Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure.</p> <p>One of the primary reasons to adopt containers is for your applications to be decoupled from the underlying environment and support higher resource utilization by “bin packing” multiple workloads onto each server. As such, the architecture of containers means that they’re deployed with multiple containers sharing the same kernel.</p> <p>The core components of the Linux kernel that are used for containers are <b>cgroups</b> — control groups, which define the resources like CPU and memory which are available to a given process — and <b>namespaces</b>, which are a way of separating processes by restricting what each process can see, so that system resources “appear” isolated to the process.</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>

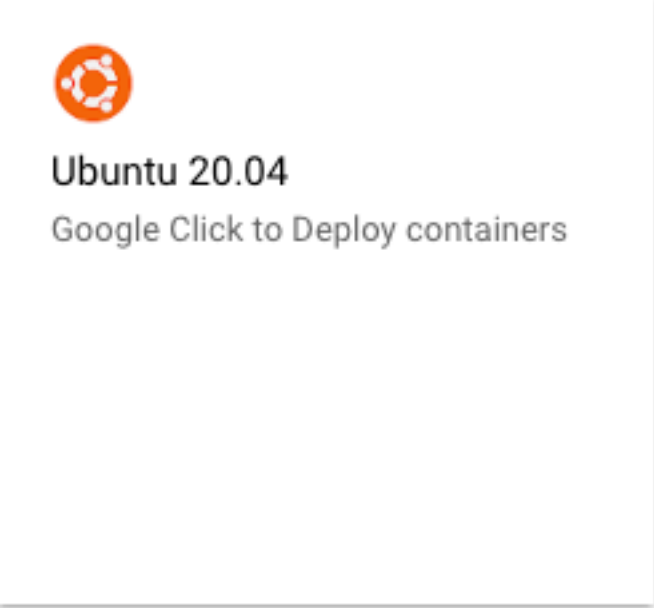
Claim 1	Accused Instrumentalities
	 <p>The diagram illustrates a container architecture stack. At the top, two light blue boxes represent 'Container 1' and 'Container 2'. Inside 'Container 1' are two yellow boxes labeled 'Application 1' and 'Libraries'. Similarly, 'Container 2' contains 'Application 2' and 'Libraries'. Below the containers is a grey box for 'Container runtime', followed by a green box for 'Host kernel', and a blue box for '(Virtualized) hardware' at the base.</p> <p><a href="https://cloud.google.com/architecture/best-practices-for-operating-containers">https://cloud.google.com/architecture/best-practices-for-operating-containers</a></p> <p>For example, Migrate to Containers automatically generates a container image, a Dockerfile for day-2 image updates and application revisions, Kubernetes deployment YAMLs and (where relevant) a persistent data volume onto which the application data files and persistent state are copied. This automated, intelligent extraction is</p> <p><a href="https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration">https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration</a></p>

Claim 1	Accused Instrumentalities
	<p>Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services.</p> <p><a href="https://cloud.google.com/learn/what-are-containers">https://cloud.google.com/learn/what-are-containers</a></p>
<p>[1d] i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications,</p>	<p>In each Accused Instrumentality, some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications.</p> <p>For example, a Docker base image serves as a self-contained unit that encompasses all the necessary components for an application to run, including the application code, runtime environment, system tools, and dependencies (i.e., SLCSEs). The images are based on existing Linux distributions, such as Debian and Ubuntu, including essential system elements (i.e., functional replicas of OSCSEs). Each container image is based on a specific base image, which contains the application code, and dependencies, including system libraries or shared library critical system elements (SLCSEs). When the container runs the image, it creates a runtime instance of that container image.</p> <p><i>See, e.g.:</i></p> <p>Many base images are basic or minimal Linux distributions: Debian, Ubuntu, Red Hat Enterprise Linux (RHEL), Rocky Linux, or Alpine. Developers can consume these images directly from Docker Hub or other sources. There are official providers along with a wide variety of other downstream repackagers that layer software to meet customer needs.</p> <p><a href="https://cloud.google.com/software-supply-chain-security/docs/base-images">https://cloud.google.com/software-supply-chain-security/docs/base-images</a></p>

Claim 1	Accused Instrumentalities
	<p data-bbox="661 293 1837 464">A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of</p> <p data-bbox="651 516 1012 792">A "container image" is your application and its dependencies, and uses a "base image" as the basis for the container image</p>  <p>The diagram illustrates the container architecture stack. At the top, three application boxes labeled 'App 1' (green), 'App 2' (yellow), and 'App 3' (pink) are shown. Below each application is a corresponding 'Bins/Libs' box in the same color. These applications and their dependencies sit on a blue 'Container Runtime' bar. Below the runtime is a white 'Host Operating System' bar, which sits on a grey 'Infrastructure' bar. The entire stack is labeled 'Containers' at the bottom.</p>

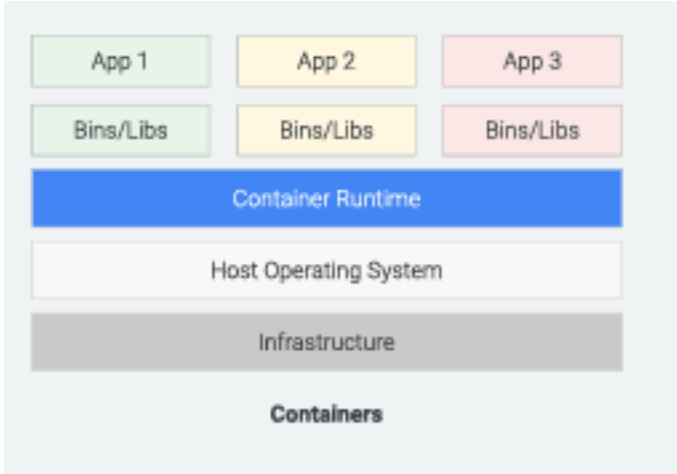
Claim 1	Accused Instrumentalities																								
	<p>The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or <b>base image</b>. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities.</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p> <table><tr><th>OS</th><th>Repository path</th><th>Google Cloud Marketplace listing</th></tr><tr><td>Debian 10 "Buster"</td><td><code>marketplace.gcr.io/google/debian10</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Debian 11 "Bullseye"</td><td><code>marketplace.gcr.io/google/debian11</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Debian 12 "Bookworm"</td><td><code>marketplace.gcr.io/google/debian12</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Rocky Linux 8</td><td><code>marketplace.gcr.io/google/rockylinux8</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Rocky Linux 9</td><td><code>marketplace.gcr.io/google/rockylinux9</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Ubuntu 20.04</td><td><code>marketplace.gcr.io/google/ubuntu2004</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr><tr><td>Ubuntu 22.04</td><td><code>marketplace.gcr.io/google/ubuntu2204</code></td><td><a href="#">Google Cloud Marketplace</a></td></tr></table> <p><a href="https://cloud.google.com/software-supply-chain-security/docs/base-images">https://cloud.google.com/software-supply-chain-security/docs/base-images</a></p>	OS	Repository path	Google Cloud Marketplace listing	Debian 10 "Buster"	<code>marketplace.gcr.io/google/debian10</code>	<a href="#">Google Cloud Marketplace</a>	Debian 11 "Bullseye"	<code>marketplace.gcr.io/google/debian11</code>	<a href="#">Google Cloud Marketplace</a>	Debian 12 "Bookworm"	<code>marketplace.gcr.io/google/debian12</code>	<a href="#">Google Cloud Marketplace</a>	Rocky Linux 8	<code>marketplace.gcr.io/google/rockylinux8</code>	<a href="#">Google Cloud Marketplace</a>	Rocky Linux 9	<code>marketplace.gcr.io/google/rockylinux9</code>	<a href="#">Google Cloud Marketplace</a>	Ubuntu 20.04	<code>marketplace.gcr.io/google/ubuntu2004</code>	<a href="#">Google Cloud Marketplace</a>	Ubuntu 22.04	<code>marketplace.gcr.io/google/ubuntu2204</code>	<a href="#">Google Cloud Marketplace</a>
OS	Repository path	Google Cloud Marketplace listing																							
Debian 10 "Buster"	<code>marketplace.gcr.io/google/debian10</code>	<a href="#">Google Cloud Marketplace</a>																							
Debian 11 "Bullseye"	<code>marketplace.gcr.io/google/debian11</code>	<a href="#">Google Cloud Marketplace</a>																							
Debian 12 "Bookworm"	<code>marketplace.gcr.io/google/debian12</code>	<a href="#">Google Cloud Marketplace</a>																							
Rocky Linux 8	<code>marketplace.gcr.io/google/rockylinux8</code>	<a href="#">Google Cloud Marketplace</a>																							
Rocky Linux 9	<code>marketplace.gcr.io/google/rockylinux9</code>	<a href="#">Google Cloud Marketplace</a>																							
Ubuntu 20.04	<code>marketplace.gcr.io/google/ubuntu2004</code>	<a href="#">Google Cloud Marketplace</a>																							
Ubuntu 22.04	<code>marketplace.gcr.io/google/ubuntu2204</code>	<a href="#">Google Cloud Marketplace</a>																							

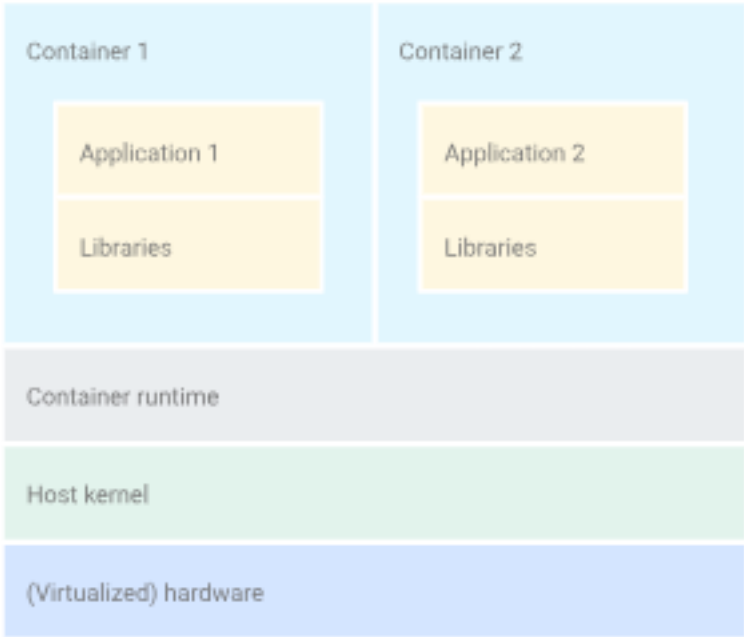
Claim 1	Accused Instrumentalities
	 <p data-bbox="690 410 1020 451"><b>Debian 10 "Buster"</b></p> <p data-bbox="690 472 1207 513">Google Click to Deploy containers</p> <p data-bbox="690 557 940 597">Open source OS</p>

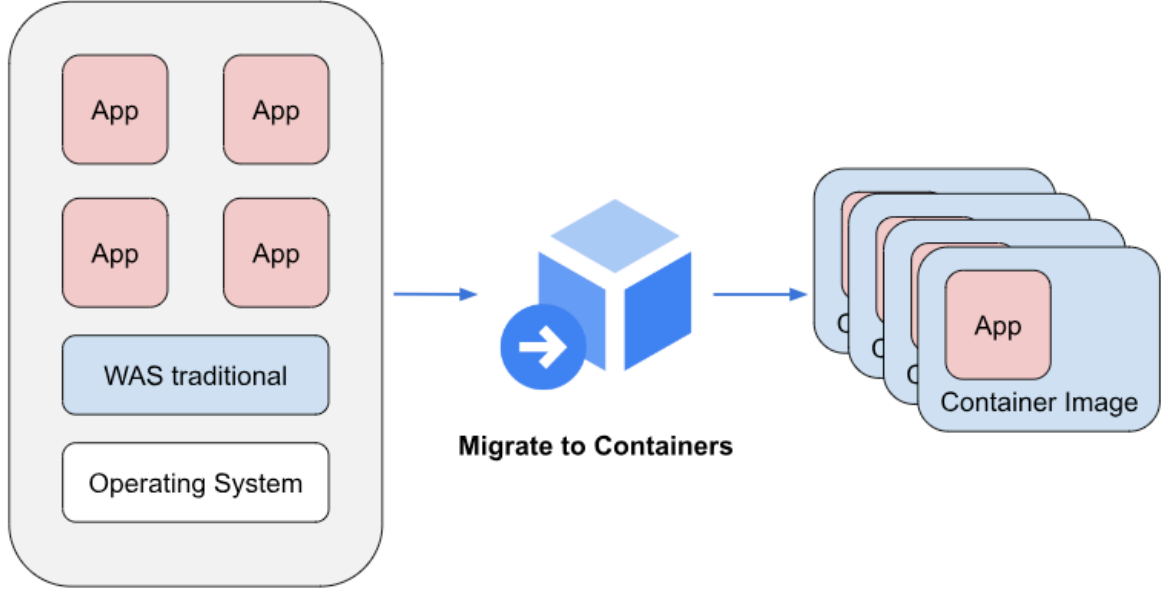
Claim 1	Accused Instrumentalities
	 <p data-bbox="636 894 1713 930"><a href="https://console.cloud.google.com/marketplace/browse?filter=solution-type:container">https://console.cloud.google.com/marketplace/browse?filter=solution-type:container</a></p>
<p data-bbox="201 954 600 1421">[1e] ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running</p>	<p data-bbox="636 954 1919 1166">In each Accused Instrumentality, an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function.</p> <p data-bbox="636 1195 1877 1406">When a Docker image is used to create a container, it creates a separate and isolated instance of a runtime environment which is independent of other containers running on the same host. Each container has its own instance of base images and its own data. The containers run in isolation, ensuring that the SLCSEs stored in the shared library are accessible to the software applications running in their respective containers. The docker image includes essential system files, libraries, and dependencies required to run the software application within the container. The Docker</p>





Claim 1	Accused Instrumentalities
<p>under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and</p>	<p>containers can share common dependencies and components using layered images. This means that multiple containers utilize the same base image to create an instance. When an instance of SLCSE is provided from the base image (i.e., from the shared library) to an individual container including application software, it operates in isolation and runs its own instance of the software application without sharing resources or critical system elements with other containers. This ensures that each container has its own isolated context. Docker containers can share common dependencies and components using layered images. This means that multiple containers can utilize the same base image. Therefore, each container, containing the application software running under the operating system, utilizes a unique instance of the corresponding critical system element to execute the respective application software for performing a same or a different function.</p> <p><i>See, e.g.:</i></p> <p>A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of</p> <p>Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure.</p>

Claim 1	Accused Instrumentalities
	<p>The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or <b>base image</b>. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities.</p>  <p>The diagram illustrates the container architecture stack. At the top, three application boxes labeled 'App 1' (green), 'App 2' (yellow), and 'App 3' (pink) are shown. Below each app is a corresponding 'Bins/Libs' box in the same color. These are all contained within a blue 'Container Runtime' box. Below the runtime is a white 'Host Operating System' box, and at the bottom is a grey 'Infrastructure' box. The entire stack is labeled 'Containers' at the bottom.</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>



Claim 1	Accused Instrumentalities
	 <p>The diagram illustrates a container architecture stack. At the top, two light blue boxes represent 'Container 1' and 'Container 2'. Inside 'Container 1' are two yellow boxes: 'Application 1' and 'Libraries'. Similarly, inside 'Container 2' are two yellow boxes: 'Application 2' and 'Libraries'. Below the containers is a grey box labeled 'Container runtime'. Underneath that is a green box labeled 'Host kernel'. At the bottom is a blue box labeled '(Virtualized) hardware'.</p> <p><a href="https://cloud.google.com/architecture/best-practices-for-operating-containers">https://cloud.google.com/architecture/best-practices-for-operating-containers</a></p>

Claim 1	Accused Instrumentalities
	 <p>The diagram illustrates a migration process. On the left, a large rounded rectangle represents a 'Websphere Application Server traditional VM'. Inside this rectangle, there are four smaller red rounded rectangles, each labeled 'App', arranged in a 2x2 grid. Below these is a blue rounded rectangle labeled 'WAS traditional', and at the bottom is a white rounded rectangle labeled 'Operating System'. An arrow points from this VM to a central icon representing a migration process. This icon consists of a blue cube with a white arrow pointing right, and a blue circle with a white arrow pointing right. Below this icon is the text 'Migrate to Containers'. Another arrow points from this icon to a stack of four blue rounded rectangles on the right. The top rectangle in the stack is labeled 'App' and 'Container Image'. Below the diagram, the text 'Apps as ibmcom/websphere-traditional container images' is displayed.</p> <p><b>Websphere Application Server traditional VM</b></p> <p><b>Migrate to Containers</b></p> <p><b>Apps as ibmcom/websphere-traditional container images</b></p> <p><a href="https://cloud.google.com/migrate/containers/docs/migrating-overview">https://cloud.google.com/migrate/containers/docs/migrating-overview</a></p>

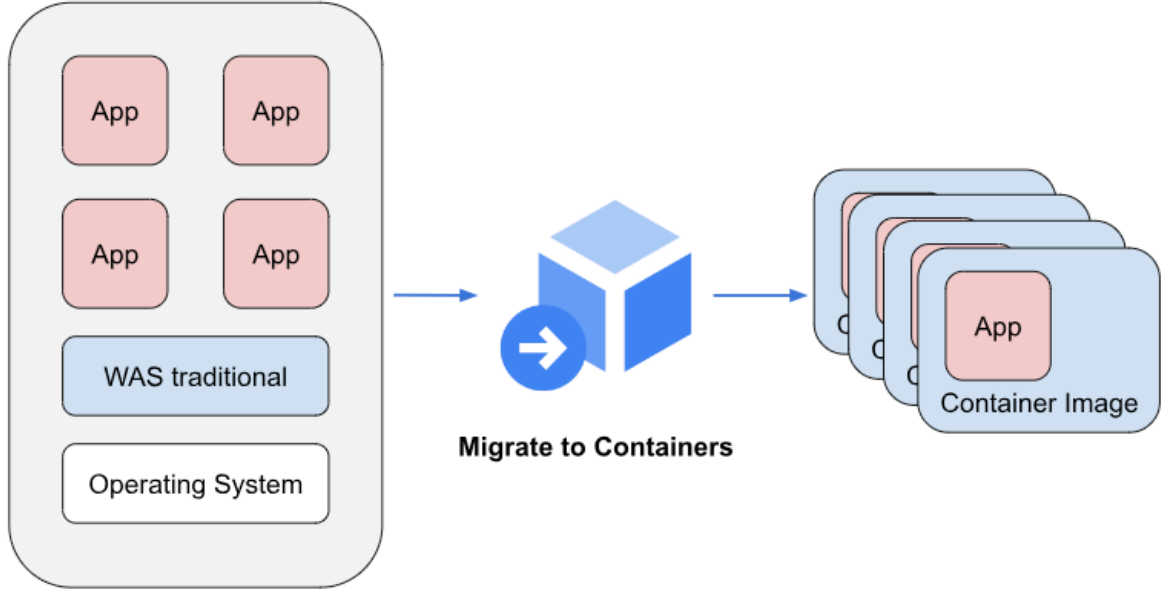
Claim 1	Accused Instrumentalities
	<div data-bbox="835 305 1054 337">Dockerfile App 1</div> <div data-bbox="699 378 1199 792"> <pre>FROM node:19.7.0</pre> <pre>ADD src_app1 /src/</pre> <pre>RUN cd /src &amp;&amp; \</pre> <pre>npm install</pre> </div> <div data-bbox="1465 305 1684 337">Dockerfile App 2</div> <div data-bbox="1325 378 1824 792"> <pre>FROM node:19.7.0</pre> <pre>ADD src_app2 /src/</pre> <pre>RUN cd /src &amp;&amp; \</pre> <pre>npm install</pre> </div> <div data-bbox="699 862 1283 976"> <p> Common layers, downloaded only once</p> <p> Layers unique to each image</p> </div> <p data-bbox="632 1065 1682 1097"><a href="https://cloud.google.com/architecture/best-practices-for-building-containers">https://cloud.google.com/architecture/best-practices-for-building-containers</a></p> <p data-bbox="653 1138 1908 1243">One method of packaging an application into a container is with the use of a Dockerfile. The Dockerfile is similar to a script which instructs the daemon on how to assemble the container image. See the <a href="#">Dockerfile reference documentation</a> for more information.</p>

Claim 1	Accused Instrumentalities
	<p>Using the Dockerfile method to build a container requires direct knowledge about the application in order to assemble the container. The first step to creating a Dockerfile is selecting an image that will be used as the basis of your image. This image should be a parent or base image maintained and published by a trusted source, usually your company.</p> <p><a href="https://codelabs.developers.google.com/developing-containers-with-dockerfiles#2">https://codelabs.developers.google.com/developing-containers-with-dockerfiles#2</a></p>
<p>[1f] iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.</p>	<p>In each Accused Instrumentality, a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.</p> <p>For example, In Docker, each container operates independently, and a Docker base image includes essential system files, libraries, and dependencies (i.e., SLCSEs) required to run the software application within the container. Based on information and belief, each element, such as system files, libraries, and dependencies (i.e., SLCSE) is associated with an execution of a predetermined function related to the application. When a Docker image is used to create a container in ECS, an instance of the SLCSE is provided to a software application. Therefore, different instances of the SLCSE are provided to different applications for performing either a same or a different function, simultaneously.</p> <p><i>See, e.g.:</i></p> <p>Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure.</p>

Claim 1	Accused Instrumentalities
	<p>The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or <b>base image</b>. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities.</p> <p><a href="https://services.google.com/fh/files/misc/why_container_security_matters.pdf">https://services.google.com/fh/files/misc/why_container_security_matters.pdf</a></p>

Claim 1	Accused Instrumentalities
	<div data-bbox="835 305 1054 337">Dockerfile App 1</div> <div data-bbox="699 378 1199 792"> <pre>FROM node:19.7.0</pre> <pre>ADD src_app1 /src/</pre> <pre>RUN cd /src &amp;&amp; \</pre> <pre>npm install</pre> </div> <div data-bbox="1465 305 1684 337">Dockerfile App 2</div> <div data-bbox="1325 378 1824 792"> <pre>FROM node:19.7.0</pre> <pre>ADD src_app2 /src/</pre> <pre>RUN cd /src &amp;&amp; \</pre> <pre>npm install</pre> </div> <div data-bbox="699 862 1283 971"> <p> Common layers, downloaded only once</p> <p> Layers unique to each image</p> </div> <p data-bbox="632 1065 1682 1097"><a href="https://cloud.google.com/architecture/best-practices-for-building-containers">https://cloud.google.com/architecture/best-practices-for-building-containers</a></p>



Claim 1	Accused Instrumentalities
	 <p>The diagram illustrates a migration process. On the left, a large rounded rectangle represents a 'Websphere Application Server traditional VM'. Inside this rectangle, there are four smaller red rounded rectangles, each labeled 'App', arranged in a 2x2 grid. Below these is a blue rounded rectangle labeled 'WAS traditional', and at the bottom is a white rounded rectangle labeled 'Operating System'. An arrow points from this VM to a central icon representing a migration process. This icon consists of a blue cube with a white arrow pointing right, and a blue circle with a white arrow pointing right. Below this icon is the text 'Migrate to Containers'. Another arrow points from this icon to a stack of four blue rounded rectangles on the right. The top rectangle in the stack is labeled 'App' and 'Container Image'. Below the stack is the text 'Apps as ibmcom/websphere-traditional container images'.</p> <p><b>Websphere Application Server traditional VM</b></p> <p><b>Migrate to Containers</b></p> <p><b>Apps as ibmcom/websphere-traditional container images</b></p> <p><a href="https://cloud.google.com/migrate/containers/docs/migrating-overview">https://cloud.google.com/migrate/containers/docs/migrating-overview</a></p>